Part 2:

IDS Performance Measurements

2.1 Intrusion Types and Intruder Personalities

2.1.1 Intruder Types

Attacks to a system are just as likely to originate from within an organization as it is to come from the outside. Denning explains that security violations could be detected from abnormal patterns of system usage [11]. This hypothesis could go a long way in contributing to the creation of a detection mechanism to be used in an Intrusion Detection System. All of the following intrusion type examples break down the behavior of an intruder in his quest to exploit a system's vulnerabilities. Knowledge-based detection methods in particular operate from this itemization of attack patterns since most of them display easily traceable symptoms.

1) *Attempted break-in* – A favorite technique used by potential attackers to gain access to a system is to first obtain a list of valid user or login names in a system. This list can easily be obtained externally by using the 'finger' command in UNIX for example. With this list, a potential attacker attempts to guess the password by keying in commonly used passwords in the hope of finding a user who is negligent. There are even programs available on the Internet to automate the process of guessing commonly used passwords when coupled with this user list [14]. Regardless of which method the attacker chooses, this act of 'guessing' would generate a symptom of an abnormally high rate of password failures either within one user name or an entire series of them.
2) *Masquerading or successful break-in* – If an external attacker is successful in guessing a user's password, he may still leave traces that are symptomatic of intruder behavior. It is very likely that an attacker will try accessing a system outside of the organization's or legitimate user's working time. This is usually to guarantee access in case the host system has limited the username to only a

single login at any given time. The fact that the intruder is using a different connection type (e.g. remote) and is accessing from a different location (e.g. outside the organization) could give the intruder away, especially if the actual user only uses a physical connection to the host system from within the organization. The behavior of an intruder might vary differently from the actual user's. While an actual user might spend most or all of his time editing a single file or working in a single area of his file system, and intruder is most likely to browse as many files and directories as possible in his curiosity. He may also arbitrarily execute system commands to see what they do [14]. Noticing that a particular user is displaying anomalous behavior or is displaying behavior that is characteristic could possibly lead to malicious intent [47] has led to many successful situations where the intruder is either caught or the intrusion is thwarted.

3) *Penetration by legitimate user* – A computer system may also experience an attack that originates locally within the confines of the organization's network itself. Suppose there is a user in the organization who desires access to files not normally permitted to him for whatever abominable purpose. The symptoms of his attempted penetration can be identified by the triggering of various protection violations whenever he tries to access unauthorized files or programs. This assumption is made on the theory that the attacker is not knowledgeable of his target's limitations and the privilege necessary to access the target [18] and that 85% of security breaches are done by insiders. Even if the user is successful in getting the file or program, his intrusion can be highlighted by the fact that he is currently accessing a file that he normally is not privy to.
4) *Leakage by legitimate user* – A much more covert and potentially damaging situation is when a legitimate user of the organization tries to leak or smuggle

one of his own work files outside the organization. Compared to the previously mentioned situation, attacks of this nature are much harder to detect since it is within the domain of a user's permissible limits [2]. The system manager must hope that the user in question tries to exhibit any elusive behavior in order to catch him in the act. This might be brought to light if the user tries to log into the system and access his resources outside of his usual times. It might also be discovered if he tries to route the stolen or leaked data to printers or file servers not normally used.

5) *Inference by legitimate user* – If a computer system hosts a database that has secure or sensitive records, a user might try to gain access to these records even though he has a lower security level. One of the ways he would attempt to circumvent the security is to try and infer, especially if the security of the database is not properly designed [26]. However, by using inference or aggregation, a user attempting to obtain unauthorized data may be flooded with more records than usual; a situation indicative of a possible intrusion.

6) *Trojan horse* – A Trojan horse is a malicious program that is planted within a system that may carry a potentially destructive payload when executed. The most common forms of Trojan horse programs nowadays may enter the organization by means of an e-mail attachment. These types of Trojan horses tempt a careless user to execute it, thus releasing the payload which might either plant a backdoor to a system or replace command files [35]. Trojan horse programs are usually written by people who intend to gain access to a system by bypassing the usual channels. Fortunately, the existence of these substituted programs can be detected in the difference of its use of CPU processing time and the amount of I/O activity it generates.

7) *Virus* – Somewhat similar to Trojan horses but far more subversive, a virus is a program that 'infects' a file and subsequently infects other files used by this infected program, thereby spreading its influence. The symptoms of its spread can be detected by a rise in disk space used by the executable file because an originally uninfected file may have a lower disk usage than an altered infected file. A rise in the frequency of executable files being rewritten might also indicate virus activity [12]. Similarly, normally unexecuted programs being run as a result of a single executable program being run could suggest virus like behavior.

8) *Denial-of-service* – A denial-of-service attack or DoS attack occurs when an attacker attempts to bring down or halt a system by exhausting it of any of its critical resources like memory, disk space, or network bandwidth. This form of attack need not originate from outside the organization as in cases normally highlighted in the media [7]. A legitimate user within the organization can cause an equal amount of damage. The beginnings of a DoS attack can be detected, however. An attacker intending to monopolize the systems services would have an unusually high activity in terms or resource usage when compared to other users connected to the system at the time. A gradual increase in resource allocation to a particular user with no indications of any kind of decrease or halting would be symptomatic of a DoS attack.

It must be emphasized, however, that portrayal of any of these behaviors alone is not an absolute and definite indication that an attack on the system or an attempt for intrusion is about to take place. Some of the scenarios mentioned previously could be valid working instances. In the example where files are being transferred to other areas of storage, this could merely mean that the user is actually shifting departments and is taking his work with him. Similarly, all other forms of software updates, workload

changes on the system, changes in job tasks and descriptions, training activities, or even simple user mistakes may display a behavior that coincidentally responds to the patterns of an attack [44]. The role of the detection method used in the IDS is to distinguish actual intruders from legitimate users and minimize the occurrences of false positive detection.

2.1.2 Intruder Personalities and Categorization

In his work presented in 1980, James P. Anderson has classified the different personalities of the intruders that could potentially threaten computer systems. All persons attempting to compromise the security of a computer system originating from outside the organization are considered to be *external penetrators*. *Internal penetrators*, on the other hand are those persons from within the organization who access data, resources, or programs to which they are not entitled to with ill intent. Internal penetrators are further subcategorized to distinguish their respective methods of system abuse.

1) The first of these subcategories are *masqueraders*. These can either be internal or external penetrators that have obtained access to another persons account. Masqueraders may have obtained access either through password theft or guessing, or may even be a user commandeering another user's workstation while he is away from his table.
2) The next subcategory is *clandestine users*. These are legitimate users of the computer system who strive to evade normal access controls on the computer system and also its auditing mechanisms in order to gain access to resources that they do not have access to.
3) The last subcategory of internal penetrators are *misfeasors*. These are users who have the authority and privilege to use the system and its resources, but abuse

their privileges through malicious activity like information leakage or indiscriminate alteration of data.

In addition to Anderson's list of intruder personalities and abuser types, automated forms of intrusions or attacks are further categorized as *malicious process* or *malware* by Brunnstein et. al. [5]. Malicious process covers all forms of non human tools of attack or intrusion like viruses, Trojan horses, data corruption software, or even programs specifically designed to cause DoS attacks on a computer system. All of these programs share the characteristic as being potentially able to introduce various undesirable consequences. These consequences include, but are not limited to, any effects that might render data or systems inaccessible, alter or destroy data, create false data, or even degrade a systems performance.

2.2 Error Types

2.2.1 Classification Errors

Intrusion Detection Systems vary in their ability to correctly distinguish a normal user from an intruder and vice-versa. If an IDS incorrectly labels a person using the system, then the IDS is said to be facing *classification errors*. Classification errors are quite literally detection mistakes done by the IDS. Some of these mistakes are inherent to the detection method in use: depending on whether the IDS relies on behavior-based or knowledge-based detection methods. There are two types of classification errors: false positive errors and false negative errors. In statistical decision theory, these errors are called Type I and Type II errors respectively [3]. This section will be devoted to describing these errors, why they occur and how to overcome them.

2.2.2 False Positive

False positive errors occur when the method used by the IDS mistakenly sees a legal user activity as illegal and therefore intrusive. More specifically, a false positive error is said to have happened when a single sequence of commands or processes produced by a legitimate operation or user is classified as anomalous, or even similar to an illegitimate operation.

False positive errors occur more commonly in behavior-based intrusion detection methods than they do in knowledge-based methods. Behavior-based methods generally detect an intrusion by scrutinizing a user's operation and checking to see if the operation conforms to a stored normal behavior pattern. In behavior-based methods, the data store or *normal* database used by the IDS contains descriptions of acceptable user or process behavior. A false positive error will occur if the IDS sees a

new sequence that is not stored in the normal database, even if the new sequence is legal. Knowledge-based intrusion detection methods on the other hand are less susceptible to false positive errors. This is because knowledge-based methods generally detect an intrusion by scrutinizing a user's operation and checking to see if the operation conforms to a known intrusion signature or behavior pattern. In knowledge-based methods, the data store or *comparisons* database used by the IDS contains data or process signatures of known intrusion techniques, or symptoms of an intrusion attack. A false positive error will only occur if the IDS somehow manages to match a sequence of user operations to that of an intrusion signature. Although this occurrence is rare, it is known to happen from time-to-time in knowledge-based detection methods [6]. More details on the susceptibility of a detection method to false positive errors will be given in Part 3, which analyzes the different detection methods used in an IDS.

In knowledge-based intrusion detection methods, false positives can be easily solved by increasing the complexity of the known intrusion signatures. This way, there would be less likelihood that a legal user operation is identical to an intrusive sequence of events. The solution is not quite as easy in behavior-based methods though. In behavior-based detection methods, false positive classification errors are caused by incomplete normal databases. In other words, the IDS was not adequately trained to see the whole range of normal and acceptable behaviors of the users or processes of a computer system. There are a couple of reasons why this type of classification error is so prevalent in behavior-based detection methods. First, an IDS that checks for anomalies usually undergo a period of non-detection, which is when the IDS has been completely deployed to a production site but detection is turned off. This is essentially the IDS "training period". During this period, the IDS scrutinizes all user and process operations and assumes that all these operations are the only acceptable behaviors of the users or processes. IDS administrators generally conduct their training period for a long

enough length of time for the IDS to get familiar with the system environment. Unfortunately, if a user is away for the entire length of that time, or if a particular system process is not used at all during that period, the IDS would see this as an anomalous sequence of events and would generate a false alarm. The second reason for false positive errors in behavior-based methods are operational changes. Changes occur quite commonly in organizations: either changes in job descriptions that would require a change in computing patterns or changes in software, such as upgrades on system processes that modify their operational behavior. If any of these changes occur after the IDS training period, it would once again see them as an anomalous sequence of events and will once again produce a false alarm [50]. Therefore, to avoid encountering false positive errors, it is not enough that the IDS administrator determine the length of time of training that the IDS needs, he must also provide rules and allowances for a user to inform him of any operational changes, so that the IDS can be trained for each individual change.

2.2.3 False Negative

False negative errors occur when the method used by the intrusion detection system mistakenly sees an illegal user activity as legal and is therefore non-intrusive. More specifically, a false negative error is said to have happened when *none* of the sequence of processes produced by an illegitimate operation or intrusion is classified as anomalous.

False negative errors occur more commonly in knowledge-based intrusion detection methods than they do in behavior-based methods. This is because knowledge-based methods depend on a repository of known intrusion technique signatures to make a positive identification. A false negative error will occur when the IDS encounters an unknown type of intrusion. In knowledge-based methods that rely

on exact specification of an intrusion, false negative errors can even occur on variations to a known intrusion technique. If the intruder modifies the intrusion pattern, or hides the sequence of intrusive events in a series of non-intrusive events, a method known as "*masking*", the IDS would also be unable to detect the intrusion, thereby causing a false negative error. Behavior-based intrusion detection methods face this problem too, though to a lesser degree. During an IDS training period, the IDS is inactive though not dormant and assumes that every single operation done during this period is a acceptable legal behavior. This unfortunately creates a window of opportunity for intrusions. If an intrusion attack occurs during this period, the IDS will actually accept the intrusion sequence as legal and normal behavior [50]. Suppose the same intrusion technique is used again after the IDS is in full operation, it would not be seen as intrusive and would pass by undetected. More details on the susceptibility of a detection method to false negative errors will be given in the Part 3, which analyzes the different detection methods used in an IDS.

In behavior-based methods, false negative classification errors can be reduced through careful control of the training period. For this purpose, the IDS administrator has the choice of either conducting the training period in a closed laboratory, where he has absolute control of the user activity and system operations that is seen by the IDS. The difficulty with this solution is that the IDS administrator must somehow generate all the normal user traffic that might be seen in a real production environment. The IDS administrator must also perform the training on a duplicate of the system that the IDS will be hosted on. This is because system processes might show different behavior on different infrastructures. A cheaper and more realistic alternative would be for the IDS administrator to deploy the IDS to the production site in secret [3]. The IDS can be trained only during normal working hours, and be put into operation outside those hours with the assumption that intrusions happen more commonly after normal operating

periods. In order for this to be effective, though, the IDS administrator must strictly enforce a security policy that prohibits after-hours operations. For knowledge-based detection methods, the only way to avoid false negative errors is to regularly update the repository of known intrusion signatures. The IDS administrator must subscribe to security advisories released by his operating system vendors and security organizations like CERT to keep abreast of intrusion activity. He can also invest in a less complex IDS that would be useful in detecting intrusion variants.

Although detection is the main priority, false positive errors have proven to be the most irritating problem faced by IDS administrators. False negatives can be reduced by adding layers of IDS defense, whereas layering will not reduce false positive rates [19]. It is very important that an intrusion detection method address these classification error issues to guarantee effective practical implementation in the workplace, where both false positive and false negative errors are reduced to a very minimum level.

2.3 Practical Considerations

Furnell and Dowland describe 3 underlying issues which can directly effect present computing practices in a particular workplace due to an implementation of an IDS [17]. Further consideration is placed on these issues with the possibility of the occurrence of false positive or false negative detection's as previously described in Section 2.2.

2.3.1 Restriction of User Activity

The first of these issues is the *restriction of user activity*. This issue is largely dependant on whether the Intrusion Detection System installed in the workplace is has *passive* or *active behavioral characteristics upon detection* of a possible intrusion. IDS that behave passively have very little in the way of restrictions to user activity. Any impediment on a users work practice would be the direct consequence of the action taken by the system managers or security officers acting upon an IDS alert. Concern arises when the IDS has an active behavioral characteristic upon detection, in other words when the program takes it upon itself to execute remedial action to halt or prevent an intrusion.

Assuming that the IDS correctly identifies the user activity as displaying intrusive tendencies or matches the signature of a known attack method, then the proactive decision taken by the IDS is justified. The only question is to what level should the IDS restrict user activity. In their architecture of a real-time IDS, Furnell and Dowland suggest that permitted behavior should be reduced in phases [17]. This is made possible through the existence of user alert status levels and activity alert status thresholds. In their research of a theoretical IDS architecture shown previously in ***Figure 1*** in Section 1.2.4, a user initially starts with a very low alert status level, for instance '1'. This allows him to access all areas of the system and execute all

commands. If, however, the user performs actions that are suspicious in nature, like repeatedly and unsuccessfully trying to access multiple directories that are outside of his group permission level, the IDS could tag the user as 'suspicious' by raising his alert level. The quantum of this alert increase would be dependent on the severity of the last suspicious behavior. In this case, let's say the IDS raises the users alert level to '5', this consequently disallows the user from performing any activity with a lower alert status threshold. The alert status threshold for all possible activities is maintained within the IDS while the threshold values themselves are maintained and updated by the system manager. If the command "copy" has a threshold of 3 and "delete" has a threshold of "1", the user will be unable to perform these actions, thus protecting the data stored in the system from theft or destruction.

The aforementioned scenario assumes that the IDS correctly identified the attack. But what if the alert was a false positive situation? What if the user was actually trying to locate a group-based file but does not have a clue about the boundaries of his group access privileges? Perhaps his subsequent task was to copy a database file necessary for his marketing presentation. However the action taken by the IDS in restricting his permitted behavior, prohibits him from continuing with his work. Should he be punished for his ineptitude? Some may argue that an IDS should be able to protect against ignorance as well to guarantee data integrity, but in real-life situations such unexpected measures would often be the cause of pressure to drop the implementation of an IDS altogether [29]. The potential for error makes this proactive approach very inappropriate in many scenarios. Ideally, system managers need to be notified immediately to investigate the validity of an imposed restriction; and as a form of prevention, users should be made to understand the proper limits of their access privilege so as not to accidentally trip any alarms.

2.3.2 Suspension of Supervision

The next issue deals largely with resource management through *suspension of supervision*. Certain detection methods used by Intrusion Detection Systems, particularly host-based detection methods require the constant transfer of audit logs to and from the system hosting the IDS [31]. Suppose that the client machines in a workplace have constant activity 24 hours a day. The continuous monitoring or user activity of a single machine connected to the IDS host would generate an unnecessary amount of system overhead in terms of disk space to store the logs and network bandwidth to transfer the audit logs to and from the client machines in a timely manner. Compound this problem with the possibility that there are dozens of other client machines, each going through a similar level of scrutiny. The strain on network and computer resources would be tremendous. Furnell and Dowland have surmised that a prolonged vigilance upon a user whose legitimacy has already been established is a waste of effort. After a reasonable amount of time analyzing a user's behavior, the IDS should have been able to safely assume that the user poses no threat to the system.

This finding is based on the theory that if the IDS is not able to detect malicious activity or signs of impostor behavior up to a certain point, then it is very unlikely that the IDS would ever detect one. Furthermore, prolonging the period of scrutiny on a user's activity might produce a higher possibility of false positives or false negatives. On one hand, the IDS might misconstrue an innocent 'disk cleanup' or 'file migration' activity by a user to be symptomatic of a damaging attacker behavior (false positive). On the other hand, the IDS might make the mistake of 'learning' a user's malicious behavior as being 'normal' behavior: such as when the user performs damaging activity often enough (false negatives).

To overcome this problem, 'crucial' monitoring periods could be configured in the Intruder Detection System by the system manager to determine the level of IDS

coverage and sensitivity [28]. The system manager should assume that during the period immediately after the start of the session, the IDS can proceed to monitor user activity to conclusively prove the authenticity of the user. Monitoring could then be terminated after a certain period on non-anomalous activity, 5 minutes for instance. If the user's machine posts any significant period of inactivity (e.g. 2-3 minutes), at which time an impostor could have potentially 'taken over' and replaced the legitimate user. The IDS could be signaled to resume analyzing user activity until user authenticity has been established again. The system manager could also decide to make the IDS resume supervision at random periods, whether there is inactivity or otherwise, to prevent the probability that a user might suddenly decide to carry out an illegal activity at any point during his work. Though this periodic suspension of supervision would lower the IDS usage frequency and no longer make it considered as a real-time IDS, it reduces waste and provides relief to the host's CPU time.

2.3.3 Response to Suspected Intrusions

An Intrusion Detection System's *response to suspected intrusions* is also an issue that needs to be fully considered in any practical implementation. As described previously in the issue of restriction of user activity, any activity or sequence of activities that is considered suspect by the IDS would trigger a defensive reaction. This action might range from the benign, simply recording the details of the activity into a anomaly log for later investigation, or reducing permitted behavior to a degree, to the more malignant, such as terminating a session or process considered to be anomalous, or locking out the user's terminal from the central system. The important thing to consider is that the IDS response to a suspected intrusion should match the severity of the suspected intrusion.

However, the assumption above may be a cause for a false sense of security. Any decision made by the IDS that allows the users to continue working, like merely recording a behavioral anomaly, would provide an intruder with more time and breathing space to cause damage. On the other side of the coin, this same assumption could also be a cause for excessive paranoia. Unless the IDS has a very high degree of certainty and affirmation, any automated decision to terminate a session or process that is suspected of displaying malicious activity is undesirable. This is because a mistake on the IDS part would itself constitute as an interruption of valid user activity, and may cause corruption of data if it were to interrupt a process in progress.

The system manager can configure the IDS to take the middle path, where the IDS issues a challenge/response query to ascertain the users identity in the event of a suspected intrusion to avoid the occurrence of a false negative. These challenges to the user's identity should be cognitive to the user so as not to waste the user's time in searching for a response. For example, the user could be asked to confirm his identity by stating his Identity Card number or Employee ID. A correct response would permit the user to proceed and reset any of the IDS doubts on the user's intentions. In the case of misfeasor activity, where the intruder might possibly be a valid user of the system, the IDS can trace this by a succession of intrusion alerts. Repeated occurrences of being asked to reconfirm his identity will erode a user's alert status level until a system lock is imposed, thereby alerting the system manager or security officer that further investigation is necessary. The limitation to this method of challenge/response would be that an IDS cannot introduce a challenge/response to a suspected malicious process that is currently in execution. A lock-down on the process may be the only option, yet this should only be implemented when the IDS has a very high degree of certainty [8].

The 3 issues mention previously, *reduction of behavior*, *suspension of supervision*, and *response to suspected intrusion*, are practical consequences that must be taken into consideration in the selection of any form of IDS. That is why an assessment on the effectiveness of the detection method in use by an IDS is prerequisite to the decision implement an IDS in a working atmosphere. The next section of this research report will outline how the effectiveness of an IDS detection method is measured.

2.4 Definition of Effectiveness

For the purposes of this research, it has been decided that measures specified by Debar et. al. is most suitable in measuring effectiveness [22]. They have outlined 5 different properties exhibited by Intrusion Detection Systems as measures to evaluate its efficiency.

1) *Accuracy* – *Accuracy* is the ability of the detection method used in the IDS to correctly label or flag actual malicious activity as intrusive or anomalous, depending on the detection method in use. A detection method is said to be *inaccurate* when it flags a legitimate user action or process activity in the environment as intrusive, thereby generating a false positive result.

2) *Performance* – The rate in which an IDS as a whole can process the contents of its audit log data determine the level of its *performance*. Real-time detection of an IDS is achieved if the IDS is able to process audit log records as soon as it is generated and completes processing in time for the next audit log record. An IDS that lags behind and processes backlogs of audit records rather than new ones is considered to have very *low performance*. This is because real-time detection is no longer possible, and alerts may only be triggered long after a perpetrator has compromised and damaged the system.

3) *Completeness* – An IDS that is able to detect all attacks in its environment is considered to have a high level of *completeness*. However, it should be noted here that an absolute level of completeness may be impossible to achieve unless the system manager has advance knowledge of how many attacks would take place within the system, such as in a testing environment. In practical use, the level of completeness can never truly be known since the number of attacks is not known beforehand. *Incompleteness* is encountered when an IDS is unable to detect an intrusion or attack to its

system. Incompleteness can only be found once the consequences, or damage, caused by an undetected attack is reported or uncovered. Even so, if the nature of the attack is an abuse of privilege, where a user gathers information he is privy to. Attacks such as these are usually never reported, therefore the completeness of a detection method might never truly be known.

4) *Fault Tolerance* - Apart from protecting its host system or environment against attacks, the IDS should itself be at least equally, if not far more highly, resistant to attacks on itself. This is the measure of the IDS ability to be *fault tolerant*. The implementation of the IDS already assumes that the host system or environment is plagued by security flaws or program vulnerabilities, the IDS should not display these flaws or vulnerabilities as well. For example, *buffer overrun* occurs when a program is fed an input string far too long for it to process, thereby halting the system. If the IDS is implemented on the host system, it should be able to identify the string entry as invalid and disallow it from passing into the system. The IDS should not face a similar problem itself when faced with the damaging string. Failure to do so means that the IDS is *not* fault tolerant.

5) *Timeliness* – Somewhat similar to *performance* but focusing on a far different scope, timeliness is a measure of the IDS ability to decide on an action as the result of the complete analysis of a set of audit log records, which was previously measured by *performance*. It is a measure of the IDS reaction time in either alerting the system manager or security officer to take further action, or in its own ability to isolate sensitive material or data from an intruder thereby circumventing any further damage. As can be inferred, this measure is highly interconnected and dependent on the performance of

the IDS as it also gauges speed of propagation of information and processing.

Velissarios and Santarossa concur with the use of these properties as a good measure of effectiveness [48]. However, they have discovered that these properties can be adjusted for optimum Intrusion Detection System efficiency. *Accuracy*, for example, can be further improved by constantly updating attack signatures to validly distinguish actual intrusions. This updating process can either be part of the security policy, or be completely automated in receiving updates from its vendor source, much like what has been done with most anti-virus programs. Consistent updates can also help improve *completeness* in a similar manner. New types of attacks are more readily identified when more signatures are loaded in the IDS.

In addition, the *performance* of a detection method can be improved by prioritizing events to be monitored and limiting the area of observation for attacks consistent with the organization's network topology and storage structure. Excluding considerably "safe" portions of the network from scrutiny would free up valuable CPU time; relieving resources for use to patrol constantly targeted areas of the network. Unfortunately, this often requires added research on the part of system managers and expertise to determine which parts of the network can actually be ignored.

Furthermore for the purposes of this research report, apart from the properties mentioned previously, the ease in handling attacks by the different intrusion types and intruder personalities as outlined in Sections 2.1 will also be a measure of how effective a particular detection method is in spotting an intrusion attempt. This is not an identical measure as the aforementioned properties. While the 5 properties outlined by Debar attempt to gauge the effectiveness of execution within the confines of a detection methods' parameters and targeted intruder types, assessing an IDS reaction to other methods of attack it was not specifically designed for would highlight the practicality of

its implementation. This exercise is necessary to reflect real life situations. An actual attacker trying to penetrate a chosen computer system would not limit himself to a single avenue of attack. Especially with the overabundance of intrusion and hacking tools freely available on the Internet, a potential intruder can fail again and again to achieve a successful break-in because time is always on his side. An Intruder Detection System on the other hand, cannot afford to fail even once. Even though it is recognized that measuring an IDS outside its boundaries would give an unfairly pessimistic view of its abilities, it is regarded as a good assessment of its potential.